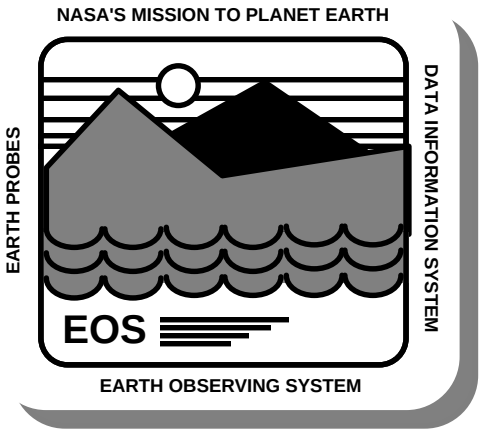
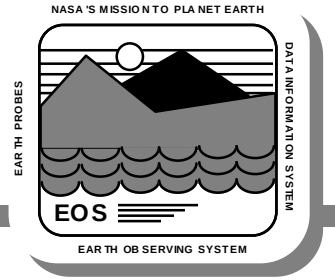


Portability Issues for ECS Science Software Integration

Narayan Prasad

Science Software Integration and Test Workshop, Landover, MD
April 18, 1995

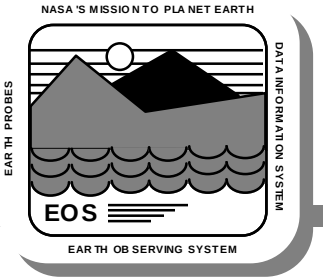
Major Incompatibilities



Due to:

- **Complexity introduced by byte ordering**
- **Word size in 32- and 64-bit machines**
  - **limitations**
  - **differences**
- **Differences in operating system**
- **Language extensions**

Byte Ordering



Architectures

- **Little-endian (Digital, MasPar, etc.) - High byte on the right**
- **Big-endian (HP, SGI, SUN, IBM, Cray, etc.) - High byte on the left**

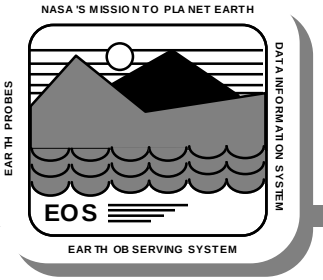
Implications

- **For high-level source code no modification is necessary; only need recompilation and relinking**
- **For data in packed binary format, need code modifications when porting across different byte-ordered architectures**
- **Binary data require byte reordering**

Compiler options

- **DEC Alpha has a switch (for Fortran only) to convert big-endian to little-endian and vice-versa**
- **No such option is available on big-endian machines to accommodate code from little-endian architectures**

Word Size



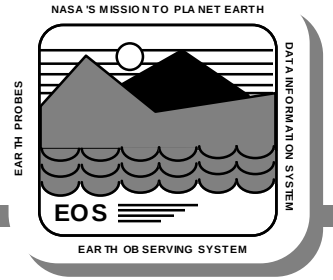
Limitations

- **Cray is word addressable (minimum is 8 bytes)**
  - **need bitwise operators manipulating sizes less than 8 bytes (can make porting difficult)**
- **SGI Power Challenge is byte addressable (minimum is 1 byte)**

Differences

- **differences in data type sizes (e.g. SGI Challenge and SGI Power Challenge)**
  - ***long int, pointer*** **(32 bits on 32-bit machines; 64 bits on 64-bit machines)**
  - ***long double*** **(64 bits on 32-bit machines; 128 bits on 64-bit machines)**
  - ***integer*, real*, complex**** **promoted internally**
  - **addresses on 64-bit machines are 64-bits**

Word Size (Cont.)



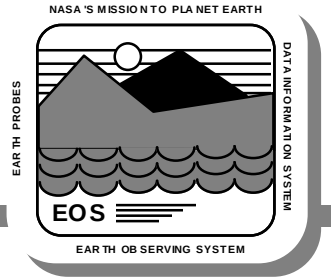
Implications

- **Run time errors**
- **Different results due to additional accuracy**
- **Integer variables that hold addresses may need to be changed**
- **Fortran-C interfaces may need modification**
- **Mixing implicit casts can give different results**
- **Adding constants (32-bit *unsigned int*) to *long***

Remedy

- **Use integer*8 to store addresses on 64-bit machines**
- **Use #if directives for bit sensitive areas of code**
- **C routines called by Fortran where variables are passed by reference must be modified to hold 64-bit addresses (no problem if *pointers* are defined)**
- **In C, always use *sizeof* to get size of variables**
- **Always define types in Fortran and cast variables in C**

Operating System



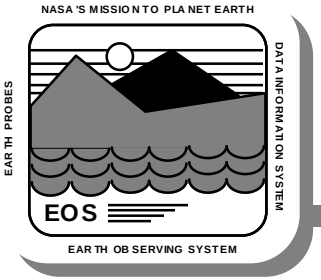
Compatibilities

- **Binary compatible (an executable created on 32-bit machine can run on 64-bit machine)**
- **Compiler switch that forces 32-bit programs to run in 32-bit mode on 64 -bit machines (at the expense of performance)**

Incompatiblities

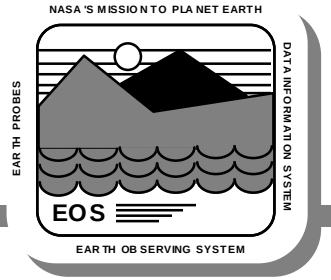
- **Cannot mix objects and Dynamic Shared Objects (DSOs) produced by 32-bit compilers with objects and DSOs produced by 64-bit compilers or vice versa**
- **Access to kernal data structures *(kmem)* must be ported to 64-bit environment**
- **IRIX GL does not yet have 64-bit support**

Language Extensions



- **Project has suggested guidelines for language extensions for C and Fortran 77**
- **It is best not to use extensions for better portability**
- **Fortran 90 interpretation may be different from that of Fortran 77 for the following situations:**
  - **Fortran 90 has more intrinsic functions and can cause name collision (could be minimal)**
  - **If a named variable is not in a common block is initialized in a DATA statement, it has the SAVE attribute in Fortran 90**
  - **Input record is padded with as many blanks as necessary to satisfy the input item and the corresponding format. Not so in Fortran 77.**
  - **Fortran 77 permits a processor to supply extra precision for a real constant when it is used to initialize a DOUBLE PRECISION data object in a DATA statement. Fortran 90 does not permit this.**

Tips For Creating Portable Programs



- **Keep portability in mind when developing code**
- **Good programming practice is essential**
- **Document areas in code that are bit sensitive (also use #if def)**
- **Avoid vendor specific extensions**
- **Don't make any assumption regarding pagesize (e.g. using *mmap()* and specifying the address). Use *getpagesize()*.**
- **Use POSIX compliant compilers fort77 and c89 during development**
- **Consult a porting and transition guide (e.g. MIPSpro 64-Bit Porting and Transition Guide from Silicon Graphics, Inc.)**
- **Early prototyping can detect problem areas**